Theoretical Computer Science 52 (1987) 77-89
North-Holland

FACTORING POLYNOMIALS AND PRIMITIVE ELEMENTS FOR SPECIAL PRIMES *

Joachim VON ZUR GATHEN
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada

Communicated by A. Schönhage
Received August 1986

Abstract. For those prime numbers p, for which all prime factors of $p-1$ are small, the two problems of finding a primitive element modulo p and of factoring univariate polynomials over finite fields of characteristic p are (deterministically) polynomial-time equivalent. Assuming the Extended Riemann Hypothesis, they can be solved in polynomial time.

1. Introduction

The problem of factoring polynomials has seen astonishing progress since 1981, culminating in probabilistic polynomial-time algorithms for factoring multivariate polynomials given by straight-line programs [23]. Within the framework of polynomial time, the major remaining theoretical challenge is to remove probabilistic choice from the algorithms. We address this challenge in the most basic case, that of factoring univariate polynomials over finite fields.

A univariate polynomial of degree d over a finite field with q elements can be factored deterministically in $(dq)^{O(1)}$ bit operations, and probabilistically (Las Vegas) in $(d \log q)^{O(1)}$ bit operations ([7, 8]; later papers on the subject include [6, 10, 11, 29]). Note that the input size is about $d \log q$, in a standard encoding. For practical purposes, the probabilistic algorithms are quite satisfactory. However, it remains a fundamental problem whether there exist deterministic methods using polynomial time, i.e., with $(d \log q)^{O(1)}$ bit operations. The current paper deals with this question, and proves that, for primes of a very special form, the factoring problem is deterministic polynomial-time equivalent to the more classical problem of finding primitive elements.

All factoring algorithms, say for multivariate polynomials over algebraic number fields or over finite fields, eventually reduce the problem to our base case of univariate polynomials over a finite field, via a modular technique. For the modular approach to factoring polynomials over Q, the deterministic algorithms over finite fields can

* Part of this work was done while the author was visiting Universität Zürich, and supported by Schweizerischer Nationalfonds, Grant 2175-0.83, and by NSERC, Grant 3-650-126-40.

be used [26]. The probabilistic polynomial-time methods for factoring (densely represented) multivariate polynomials over finite fields of characteristic p [12, 18, 25] can be made deterministic if and only if the same is true for the univariate case over Z_p.

The paper is organized as follows. In Section 2 we will present some results concerning deterministic polynomial-time factorization. We will prove that for our question only the size of the characteristic p, not the actual size of the field, is relevant. In particular, over finite fields of characteristic 2 (or any fixed p) one can factor deterministically in polynomial time. In fact, the general problem is reducible to the special case of polynomials which are the product of linear factors over the prime field. The factorization pattern—consisting of the degrees and multiplicities of the irreducible factors—can be computed in deterministic polynomial time, also for large characteristic.

The subsequent results all deal with the special type of prime numbers p, where $p-1$ is 'smooth', i.e., has only small prime factors. Then the group $Z_p^\times$ of units modulo p is a product of cyclic q-groups for small primes q. More precisely, we define the smoothness $S(k)$ of an integer k to be the largest prime number dividing k. Then, starting in Section 3, 'polynomial time' will mean a number of bit operations that is polynomial in the input size plus $S(p-1)$. In Section 3, we will give a deterministic polynomial time algorithm for the special factorization problem of finding qth roots, where q divides $p-1$, and a primitive element modulo p is given.

Then we will consider the two computational problems of finding a primitive element modulo p, and of factoring a univariate polynomial over a finite field of characteristic p. We will show that these two problems are polynomial-time equivalent, via Cook-reductions. In Section 4, we will reduce factoring to primitive elements, and in Section 5 present the (easier) reduction in the other direction. In the last paragraphs of the paper, we discuss parallel algorithms. Assuming the Extended Riemann Hypothesis (ERH), primitive elements can be found in time polynomial in $\log p + S(p-1)$ [3, 4, 33]. Therefore, also the present factoring problem can be solved deterministically in polynomial time, under the ERH.

In spite of the latter 'positive' result, the equivalence in a special case seems to indicate that the general deterministic polynomial-time factoring problem may be similarly hard as the long-standing open question of finding primitive elements fast.

The apparent difficulty of the factoring question is also illustrated by the fact that even for quadratic polynomials, the best deterministic result to date is Schoof's [32] method for finding modular square roots of small integers. Huang [22] shows that cyclotomic polynomials can be factored deterministically in polynomial time. Both these results assume the ERH. Adleman and Lenstra [2] and Von Zur Gathen [16] discuss the problem of finding irreducible polynomials over finite fields in deterministic polynomial time.

Moenck [27] had shown that polynomials in $Z_p[x]$ can be factored in deterministic polynomial time if $p-1 = L \cdot 2^l$ with L of order $\log p$ and a primitive element modulo p given. The present paper generalizes this result. (It is interesting to note that

Moenck considers the (probabilistic) generation of a primitive element as an easy problem; in fact, we will show here that it is (polynomial-time) equally hard as factoring polynomials.)

It follows from [1] that there exist non-uniform families of polynomial-size deterministic Boolean circuits for factoring polynomials over finite fields. Camion [10] provides a specific algorithm of this type, which factors polynomials from $Z_p[x]$ of degree d in $(d \log p)^{O(1)}$ steps. A description of his algorithm can be generated in $(\log p)^{O(1)}$ steps by a probabilistic algorithm, but it is not clear how to do this deterministically.

Two ways of generalizing the present result suggest themselves:

(1) Does it help in the factorization problem to know the prime factors of $p-1$, even when they are large? (The known polynomial-time methods for testing whether a given integer is primitive modulo p assume knowledge of these prime factors.)

(2) In the present approach, can one replace properties of $p-1$ by properties of $p+1$, or of other cyclotomic polynomials in p? (Bach and Shallit [5] give such generalizations for the harder problem of factoring integers.)

2. Special factorization problems with deterministic polynomial-time solutions

In this section, we will collect some results, mainly from the literature, on special cases of the factorization problem which have deterministic polynomial-time algorithms.

For the factoring problem, we have as input a prime number p, an irreducible monic polynomial $w \in Z_p[y]$ of degree l, and a monic polynomial $f \in F[x]$ of degree d, where $F = Z_p[y]/(w) = \mathrm{GF}(p^l)$. Output is a factorization $(f_1, d_1; \ldots; f_s, d_s)$ of f, with $f_1, \ldots, f_s \in F[x]$ distinct irreducible monic polynomials, $d_1, \ldots, d_s \geq 1$ and $f = f_1^{d_1} \cdots f_s^{d_s}$. We assume some standard encoding: an integer is given by its binary representation, an element of Z_p by an integer between 0 and $p-1$, an element $u \in F$ by elements $u_0, \ldots, u_{l-1}$ of Z_p such that $u = \sum_{0 \leq i < l} u_i y^i \bmod w$, and a polynomial in $F[x]$ by its coefficient sequence. Then the input size for the factoring problem is about $dl \log p$.

We first remark that the factorization can be computed deterministically with cost polynomial in p without assuming $p-1$ to be smooth. This is based on a modification of Berlekamp's [8] algorithm proposed by Cantor and Zassenhaus [11]; the latter authors are only interested in probabilistic algorithms. In the special case of characteristic 2, e.g., our algorithm is deterministic and conceptually simpler than theirs.

We consider $R = F[x]/(f)$ as an ld-dimensional vector space over Z_p, with basis $\{x^i y^j \bmod (w, f): 0 \leq i < d, 0 \leq j < l\} \subseteq R$ and residue class mapping $g \to \bar{g}$ from $F[x]$ to R. We compute $g_1 = 1, g_2, \ldots, g_s \in F[x]$ of degree less than d such that $\bar{g}_1, \ldots, \bar{g}_s$ form a basis of the Z_p-vector space

$$B = \{u \in R: u^p = u\} \subseteq R,$$

the 'Berlekamp subalgebra' of R. The following is well-known.

Fact 2.1. (i) *s is the number of distinct irreducible monic factors of f.*

(ii) *f irreducible $\Leftrightarrow s=1$.*

(iii) *If $\beta=(\beta_1,\ldots,\beta_s): R \to^{\cong} R_1\times\cdots\times R_s$ is the Wedderburn decomposition of R, and $R_k = F[x]/(f_k^{d_k})$, then Z_p is contained in R_k in a natural way, and*

$$\beta(B) = Z_p\times\cdots\times Z_p.$$

(iv) *If $g\in F[x]$ satisfies $\beta_k(\bar g)=0$ and $\beta_l(\bar g)\neq 0$ for some $k, l\leqslant s$, then $\gcd(f,g)$ is a nontrivial factor of f. We call such a g a 'factoring polynomial for f'.*

(v) *$Z_p\subseteq B\subseteq R$ is mapped via β to the diagonal of $Z_p\times\cdots\times Z_p$. If $\bar g\in B\backslash Z_p$, then there exists a $u\in Z_p$ such that $g+u$ is a factoring polynomial for f.*

For proofs, see [9] for (i), [7] (for $F=Z_p$), [11, 14].

If we are given $g\in F[x]$ with $\bar g\in B\backslash Z_p$ and compute the p polynomials $\gcd(f, g+u)$ for $u\in Z_p$, then Fact 2.1(v) guarantees that we find a factor of f. For a complete factorization, we can assume that f is monic and square-free, and build up finer and finer factorizations $S\subseteq F[x]\backslash F$ consisting of monic polynomials with $\prod_{h\in S} = f$. Initially, $S=\{f\}$. For $i=1,\ldots,s$ and $\alpha\in F$, we compute, for each $h\in S$, $g_i-\alpha \bmod h$ and $u=\gcd(h, g_i-\alpha)$, and replace S by $(S\backslash\{h\})\cup\{h/u, u\}$ if $u\neq 1, h$. The final S contains precisely the irreducible factors of f.

Let $M(d)$ be such that polynomials of degree at most d over a finite field of characteristic p can be multiplied with $O(M(d))$ operations in the field and similarly d^ω for multiplication of $d\times d$-matrices.

Theorem 2.2. *Polynomials of degree d over $\mathrm{GF}(p^l)$ can be factored deterministically with $O((dl)^\omega + (l\log p + p\log d)\cdot dM(d)M(l))$ operations in Z_p.*

With $M(d)=O(d\log d\log\log d)$ [30] and $\omega<2.4$ [13a] the resulting estimate $O(p(dl)^\omega)$ improves the results in the literature. The case of characteristic 2 is particularly important, e.g., in algebraic coding theory. Camion [10] proves existence of a polynomial-time factoring procedure, without exhibiting one. The specific algorithm indicated above works in polynomial time.

Corollary 2.3. *Polynomials of degree d over $\mathrm{GF}(2^l)$ can be factored deterministically with $O((dl)^\omega)$ operations in Z_p.*

As observed in [8], it is sufficient to find a factor of $\mathrm{res}(f, g+t)\in Z_p[t]$ in order to factor f, using Fact 2.1(v), where $\bar g\in B\backslash Z_p$. The resultant has coefficients from Z_p since

$$\mathrm{res}(f, g+u)=0 \Leftrightarrow \deg(\gcd(f, g+u))\geqslant 1$$

$$\Leftrightarrow \exists k\leqslant s\ \beta_k(\bar g+u)=0 \Rightarrow \exists k\ u=-\beta_k(\bar g)\in Z_p$$

for any u in an algebraic closure of F. Since the square-free part of polynomials can be computed in polynomial time, these observations yield the following result.

Theorem 2.4. *Factoring univariate polynomials over finite fields of characteristic p is polynomial-time reducible to factoring univariate square-free polynomials with only linear factors over* Z_p.

If a monic polynomial $f \in F[x]$ has a factorization $f = f_1^{d_1} \cdots f_s^{d_s}$ with $f_1, \ldots, f_s \in F[x]$ pairwise distinct monic irreducible, and $d_1, \ldots, d_s \geq 1$, then we call $(\deg f_1, d_1; \cdots; \deg f_s, d_s)$ a *factorization pattern* of f. The well-known techniques of computing the square-free decomposition and the 'distinct-degree factorization' [24, Section 4.6.2] yield the following result, which also follows from [20].

Theorem 2.5. *The factorization pattern of univariate polynomials over finite fields can be computed deterministically in polynomial time.*

3. Primes p with $p-1$ smooth

Throughout the paper, n denotes an input size parameter. Given a prime $p < 2^n$, one can check in polynomial time whether $p-1$ has only prime factors $\leq n$, and in the affirmative case, compute the prime factorization of $p-1$.

We consider the following two computational problems.

PRIMITIVE$_n$: Input is a prime number $p < 2^n$ with $S(p-1) \leq n$. (Recall that $S(p-1) = \max\{p_1, \ldots, p_r\}$ if $p_1, \ldots, p_r$ are the prime factors of $p-1$.) Output is some $a \in N$ such that $2 \leq a < p$, and $a \bmod p$ is a primitive element modulo p.

Note that the output is not uniquely determined. Formally, PRIMITIVE$_n$ is the set of all Boolean functions g such that g(binary representation of p) is the binary representation of a primitive element modulo p if $S(p-1) \leq n$. An algorithm computes it if it computes one of these functions. It can artificially be made unique by stipulating, e.g., that the output be the smallest positive primitive element.

FACTOR$_n$ (Factoring polynomials of degree less than n over fields with at most 2^{n^2} elements): Inputs are p, w, f as in Section 2 with $p < 2^n$ and $l, d < n$, and furthermore $S(p-1) \leq n$. Output is a factorization $(f_1, d_1; \ldots; f_s, d_s)$ of f.

Again, the output is not uniquely determined; we can permute the (f_k, d_k)'s. It can be made unique by stipulating some lexicographic order on the integers representing the output.

In a standard encoding the input and output size for PRIMITIVE$_n$ is at most n, and for FACTOR$_n$ at most n^3.

For the remainder of the paper we fix the above notation. In particular, 'polynomial time' always means a number of bit operations that is polynomial in n, and the inputs are as above.

In this section, we note two consequences of having a primitive element a mod p: an efficiently computable decomposition α of $\mathbf{Z}_p^\times$, and extracting p_ith roots, where p_i is a prime divisor of $p-1$. This immediately generalizes to qth roots for any divisor q of $p-1$.

Lemma 3.1. *Let $p-1=p_1^{e_1}\cdots p_r^{e_r}$ be the prime factorization of $p-1$ so that $S(p-1)=\max\{p_1,\ldots,p_r\}$, and let a be a primitive element modulo p.*

(1) *There exists exactly one isomorphism*

$$\alpha=(\alpha_1,\ldots,\alpha_r):\mathbf{Z}_p^\times\to\mathbf{Z}_{p_1^{e_1}}\times\cdots\times\mathbf{Z}_{p_r^{e_r}}$$

with $\alpha(a)=(1,\ldots,1)$. For any $b\in\mathbf{Z}_p^\times$, $\alpha(b)$ can be computed with $O(\log^{3+\varepsilon}(p)\cdot S(p-1))$ bit operations, for any $\varepsilon>0$.

(2) *For any $b\in\mathbf{Z}_p^\times$ and $1\leq i\leq r$, the following are equivalent:*

(i) *b has a p_i-th root,*

(ii) $\alpha_i(b)\equiv 0 \bmod p_i$,

(iii) $b^{(p-1)/p_i}=1$.

If the conditions are satisfied, then the p_i distinct roots can be found at the same cost as in (1).

Proof. (1): Given $b\in\mathbf{Z}_p^\times$ and $1\leq i\leq r$, we define

$$\alpha_i(b)=u_{i0}+u_{i1}p_i+\cdots+u_{i,e_i-1}p_i^{e_i-1} \bmod p_i^{e_i}$$

with $0\leq u_{it}<p_i$ inductively for $t=0,\ldots,e_i-1$ by

$$(b\cdot a^{-(u_{i0}+\cdots+u_{it}p_i^t)})^{(p-1)/p_i^{t+1}}=1.$$

Given $u_{i0},\ldots,u_{it-1}$, we find by exhaustive search the unique u_{it} with $0\leq u_{it}<p_i$ satisfying this relation. This provides a computation for α; $\alpha(a)=(1,\ldots,1)$ and uniqueness of α are clear. Each $\alpha_i(b)$ can be calculated in $O(e_ip_i\log p)$ arithmetic operations modulo p, and one such operation in $O(\log p\log\log p\log\log\log p)$ bit operations [31], which we have abbreviated to $O(\log^{1+\varepsilon}(p))$. For the final estimate, we use that $\sum_{1\leq i\leq r}e_ip_i\leq\log p\cdot S(p-1)$.

(2): Let $b=a^k$ with $0\leq k\leq p-2$. Then

$$\begin{aligned} b \text{ has a } p_i\text{th root} &\Leftrightarrow \exists c\in\mathbf{Z}_p^\times\ c^{p_i}=b \\ &\Leftrightarrow p_i \text{ divides } k \Leftrightarrow \alpha_i(b)\equiv 0 \bmod p_i \\ &\Leftrightarrow (p-1)/p_i\cdot k\equiv 0 \bmod p-1 \Leftrightarrow b^{(p-1)/p_i}=1. \end{aligned}$$

Since $b\neq 0$, the polynomial $x^{p_i}-b\in\mathbf{Z}_p[x]$ is separable, and has p_i distinct roots. To compute the roots $c_0,\ldots,c_{p_i-1}$, find $\alpha(b)$ and

$$c_u=\alpha^{-1}\left(\frac{\alpha(b)}{p_i}+(0,\ldots,0,up_i^{e_i-1},0,\ldots,0)\right)$$

for $0\leq u<p_i$. Here, the jth component v_j of $\alpha(b)/p_i$ satisfies $p_iv_j=\alpha_j(b)$. For $w_1,\ldots,w_r\in\mathbf{Z}$ we have

$$\alpha^{-1}((w_1 \bmod p_1^{e_1},\ldots,w_r \bmod p_r^{e_r}))=\prod_{1\leq i\leq r}a^{w_i(p-1)/p_i^{e_i}}. \qquad \square$$

Remark 3.2. It would be interesting to replace the exhaustive search for u_{it} by a calculation polynomial in $\log p$. Finding p_ith roots deterministically in polynomial time would be a step towards replacing the condition '$p-1$ has small prime factors' by 'the factorization of $p-1$ is known'; one would still have to deal with Step 6 of the algorithm in Section 4. However, even the special factorization problem of root-finding is not easy; Schoof [32] has a deterministic polynomial-time algorithm for finding modular square roots of small integers, assuming the ERH.

4. Reducing factoring to primitive elements

In this section, the technical core of the paper, we present a deterministic polynomial-time reduction of factoring to the problem of finding primitive elements, for our special type of prime. The main result is the following theorem.

Theorem 4.1. *On input $f \in \mathrm{GF}(p^l)[x]$ of degree d and a primitive element modulo p, the algorithm to be described below can be executed with*

$$O([(dl)^{1+\varepsilon} \cdot ((dl)^{2.4} + \log^2 p) + S(p-1) \cdot (\log^2 p + l^{1+\varepsilon} \log p + (dl)^{1+\varepsilon})] \cdot \log^{1+\varepsilon} p)$$

bit operations for any $\varepsilon > 0$, or $O(n^8)$ bit operations, where $S(p-1)$ is the largest prime factor of $p-1$, and $n = \max\{d, l, \log p, S(p-1)\}$. If f is reducible, the algorithm returns a nontrivial factor of f.

Note that the input size is about $dl \log p$, or n^3.

Corollary 4.2. FACTOR *is polynomial-time (Cook-)reducible to* PRIMITIVE.

Here FACTOR denotes the infinite family obtained by combining all the functions in FACTOR_n, for $n \in N$; similarly for PRIMITIVE.

Proof of Theorem 4.1. To prove the theorem, we use the notation of Sections 2 and 3, and start with the two isomorphisms

$$\alpha = (\alpha_1, \ldots, \alpha_r) : Z_p^\times \to Z_{p_1^{e_1}} \times \cdots \times Z_{p_r^{e_r}},$$

$$\beta = (\beta_1, \ldots, \beta_s) : B \to Z_p \times \cdots \times Z_p = Z_p^s,$$

$$\beta_k(\bar{g}) = g \bmod f_k^{d_k}.$$

Together they provide a matrix decomposition of the multiplicative group $B^\times$ of units in B:

$$\gamma = (\gamma_{ik})_{1 \le i \le r, 1 \le k \le s} : B^\times \to (Z_p^\times)^s \to \prod_{\substack{1 \le i \le r \\ 1 \le k \le s}} Z_{p_i^{e_i}}$$

with

$$\gamma_{ik}(\bar{g}) = \alpha_i(\beta_k(\bar{g})).$$

By writing an element $u \in \mathbf{Z}_{p_i^{e_i}}$ in p_i-adic notation

$$u \equiv u_0 + u_1 p_i + \cdots + u_{e_i-1} p_i^{e_i-1} \bmod p_i^{e_i}, \qquad 0 \leq u_{it} < p_i,$$

we obtain a further decomposition

$$\delta = (\delta_{ikt})_{\substack{1 \leq i \leq r \\ 1 \leq k \leq s \\ 0 \leq t < e_i}} : B^\times \to \prod_{\substack{1 \leq i \leq r \\ 1 \leq k \leq s \\ 0 \leq t < e_i}} \{0, \ldots, p_i - 1\}$$

of the set $B^\times$ (δ is not a group homomorphism). Our goal is to find a factoring polynomial h_4 for f. We start with an arbitrary polynomial g such that $\bar{g} \in B^\times \backslash \mathbf{Z}_p$, and compute h_1, h_2, h_3 having more and more entries in the δ-array equal to zero: $\delta_{ikt}(h_1)$ is nonzero for only one value of i, $\delta_{ikt}(h_2)$ is nonzero for only one value of i and $t = e_i - 1$, and h_3 has these properties, and also $\delta_{*k*}(h_3) = 0$ and $\delta_{*l*}(h_3) \neq 0$ for some $k, l \leq s$. Finally, $h_4 = h_3 - 1$. The algorithm is described formally at the end of this section.

We can assume $s \geq 2$ and $g \in F[x]$ with $\bar{g} \in B \backslash \mathbf{Z}_p$. Furthermore we can assume that $\bar{g} \in B^\times$ is a unit since otherwise $\gcd(f, g)$ is a nontrivial factor of f. There exist $k, l \leq s$ such that $\beta_k(\bar{g}) \neq \beta_l(\bar{g})$, and also some $i \leq r$ such that

$$\gamma_{ik}(\bar{g}) \neq \gamma_{il}(\bar{g}).$$

We write $q_i = (p-1)/p_i^{e_i}$ and, for any $j \leq r$, consider the vector $\gamma_{j*}(\bar{g}^{q_i}) \in (\mathbf{Z}_{p_j^{e_j}})^s$. Then,

$$i \neq j \Rightarrow \gamma_{j*}(\bar{g}^{q_i}) = 0, \tag{1}$$

$$\gamma_{i*}(\bar{g}) \text{ nonconstant} \Leftrightarrow \gamma_{i*}(\bar{g}^{q_i}) \text{ nonconstant}. \tag{2}$$

Thus we compute g^{q_i} for $i = 1, \ldots, r$ and $h_1 \in F[x]$ of degree less than d such that $\bar{h}_1 = \bar{g}^{q_i}$ for some i with $\bar{g}^{q_i} \notin \mathbf{Z}_p$. We choose such an i, which exists by the above reasoning.

Now we consider

$$\gamma_{ik}(\bar{h}_1) \equiv \sum_{0 \leq t < e_i} \delta_{ikt}(\bar{h}_1) p_i^t \bmod p_i^{e_i}.$$

We know that the vectors $(\delta_{ik0}(\bar{h}_1), \ldots, \delta_{ik,e_i-1}(\bar{h}_1))$ are not identical for all $k \leq s$. We consider the 'lower common part' $(u_0, \ldots, u_{m-1})$ of these vectors, defined by $0 \leq u_t < p_i$ and

$$\forall k \leq s \; \forall t < m \quad \delta_{ikt}(\bar{h}_1) = u_t,$$

$$\exists k, l \leq s \quad \delta_{ikm}(\bar{h}_1) \neq \delta_{ilm}(\bar{h}_1).$$

In order to compute this m, note that

$$\bar{h}_1^{p_i^{e_i-m}} = \alpha^{-1}(0, \ldots, 0, p_i^{e_i-m} \cdot (u_0 + \cdots + u_{m-1} p_i^{m-1}) \bmod p_i^{e_i}, 0, \ldots, 0) \in \mathbf{Z}_p^\times,$$

$$\bar{h}_1^{p_i^{e_i-m-1}} \notin \mathbf{Z}_p.$$

Thus, m equals the largest number t such that $\bar{h}_1^{p_i^{e_i-t}} \in \mathbf{Z}_p^\times$. Then $0 \leq m < e_i$, and $b = \bar{h}_1^{p_i^{e_i-m}} \in \mathbf{Z}_p^\times$ has a p_ith root, say $c \in \mathbf{Z}_p^\times$, by Lemma 3.1. Thus,

$$\alpha_i(c) \equiv p_i^{e_i-m-1} \cdot (u_0 + \cdots + u_{m-1} p_i^{m-1} + u p_i^m) \bmod p_i^{e_i} \quad \text{for some } u \in \mathbf{Z}$$

and

$$\alpha_j(c) = 0 \quad \text{for } j \neq i, \text{ by (1).} \tag{3}$$

Compute $h_2 \in F[x]$ of degree less than d such that

$$\bar{h}_2 = \bar{h}_1^{p_i^{e_i-m-1}} c^{-1} \in B.$$

Then

$$\gamma_{j*}(\bar{h}_2) = 0 \quad \text{for } j \neq i, \text{ by (1) and (3),} \tag{4}$$

$$\forall k \leq s \quad \gamma_{ik}(\bar{h}_2) = p_i^{e_i-m-1} \cdot \gamma_{ik}(\bar{h}_1) - \alpha_i(c)$$

$$\equiv p_i^{e_i-m-1} \sum_{0 \leq t < e_i} \delta_{ikt}(\bar{h}_1) p_i^t - p_i^{e_i-m-1} \cdot \left(\sum_{0 \leq t < m} u_t p_i^t + u p_i^m \right)$$

$$\equiv (\delta_{ikm}(\bar{h}_1) - u) p_i^{e_i-1} \bmod p_i^{e_i}.$$

In particular, there exist $k, l \leq s$ such that $\gamma_{ik}(\bar{h}_2) \neq \gamma_{il}(\bar{h}_2)$. Choose some such k, l, and let $0 \leq v < p_i$ be such that

$$vq_i \equiv -\delta_{ikm}(\bar{h}_1) + u \bmod p_i,$$

where $q_i = (p-1)/p_i^{e_i}$. Now we set $h_3 = a^{v(p-1)/p_i} h_2 \in F[x]$. Then,

$$\gamma_{j*}(\bar{h}_3) = 0 \quad \text{for } j \neq i, \text{ by (4),}$$

$$\gamma_{ik}(\bar{h}_3) = (v(p-1)/p_i) \cdot \alpha_i(a) + \gamma_{ik}(\bar{h}_2)$$

$$\equiv vq_i p_i^{e_i-1} + (\delta_{ikm}(\bar{h}_1) - u) p_i^{e_i-1} \equiv 0 \bmod p_i^{e_i},$$

$$\gamma_{il}(\bar{h}_3) \equiv vq_i p_i^{e_i-1} + (\delta_{ilm}(\bar{h}_1) - u) p_i^{e_i-1} \not\equiv 0 \bmod p_i^{e_i}.$$

Therefore, $\alpha_j(\beta_k(\bar{h}_3)) = 0$ for all j, and $\alpha_i(\beta_l(\bar{h}_3)) \neq 0$. Thus, $\beta_k(\bar{h}_3) = 1$ and $\beta_l(\bar{h}_3) \neq 1$, and $h_4 = h_3 - 1$ is a factoring polynomial.

In the above discussion we have not distinguished between what we can do computationally ('choose $i \leq r$') and not ('choose $k, l \leq s$'). The following algorithm results from the discussion. We assume that we have a basis $\bar{g}_1 = 1, \ldots, \bar{g}_s$ of B, as at the beginning of the section, and a primitive element a modulo p.

Step 1: If $s = 1$, return 'f is irreducible' and stop. Else set $g = g_2$. If $\gcd(f, g) \neq 1$, return this nontrivial factor of f and stop.

Step 2: For $1 \leq j \leq r$, compute $w_j \in F[x]$ of degree less than d such that

$$w_j \equiv g^{q_j} \bmod f,$$

with $q_j = (p-1)/p_j^{e_j}$. Let i be the first value of j such that $w_j \notin \mathbb{Z}_p$, and set $h_1 = w_i$.

Step 3: For $t = 0, \ldots, e_i - 1$, compute $y_t \in F[x]$ of degree less than d such that

$$y_t \equiv h_1^{p_i^{e_i-t}} \bmod f.$$

Set

$$m = \max\{t : y_t \in \mathbb{Z}_p\}.$$

Step 4: Compute $c \in \mathbb{Z}_p^\times$ such that $c^{p_i} = y_m$.

Step 5: Compute $h_2 \in F[x]$ of degree less than d such that

$$h_2 \equiv h_1^{p_i^{e_i-m-1}} c^{-1} \bmod f.$$

Step 6: For $0 \leq v < p_i$ compute

$$z_v = a^{v(p-1)/p_i} h_2 - 1 \in F[x].$$

Return $\gcd(f, z_v)$ if it is nontrivial, and stop.

Correctness of the algorithm has been proved above. The dominating computing times are as follows.

Step 0 (Computing $g_1, \ldots, g_s$): $O((dl)^{2.4} + dl \log p)$ operations in R, using fast matrix arithmetic [13a];

Steps 2, 3, 5: $O(\log^2 p)$ operations in R;

Step 4: $O(\log^2 p \cdot S(p-1))$ operations in $\mathbf{Z}_p$;

Step 6: $O(S(p-1)(\log p + d^{1+\varepsilon}))$ operations in F, for any $\varepsilon > 0$.

With fast integer and polynomial arithmetic, the time estimate of Theorem 4.1 follows. □

5. Reduction of primitive elements to factoring

As usual, let p be a prime number, $p-1 = p_1^{e_1} \cdots p_r^{e_r}$, and $a \in \mathbf{Z}_p^\times$. The following are equivalent:

$$a \text{ is a primitive element modulo } p, \tag{5}$$

$$\mathbf{Z}_p^\times = \{a^i : 0 \leq i \leq p-2\}, \tag{6}$$

$$\forall i \leq r \quad a^{(p-1)/p_i} \neq 1, \tag{7}$$

$$\forall i \leq r \quad a \text{ has no } p_i\text{th root in } \mathbf{Z}_p^\times. \tag{8}$$

If $b_1, \ldots, b_r \in \mathbf{Z}_p^\times$ are such that $b_i^{p_i^{e_i}} = 1$ and $b_i^{p_i^{e_i-1}} \neq 1$, then $a = b_1 \cdots b_r$ is a primitive element modulo p. To check (7), let $1 \leq i \leq r$. Then $c_i = b_i^{p_i^{e_i-1}}$ has order p_i, and thus $c_i^{q_i} \neq 1$, where $q_i = (p-1)/p_i^{e_i}$. Hence,

$$a^{(p-1)/p_i} = \prod_{1 \leq j \leq r} b_j^{(p-1)/p_i} = b_i^{(p-1)/p_i} = c_i^{q_i} \neq 1.$$

Theorem 5.1. FACTOR *and* PRIMITIVE *are polynomial-time equivalent.*

Proof. By Corollary 4.2, it is sufficient to reduce PRIMITIVE to FACTOR. Let n be an input size parameter, $p < 2^n$ a prime, $p-1 = p_1^{e_1} \cdots p_r^{e_r}$ with $p_1, \ldots, p_r \leq n$, and $1 \leq i \leq r$. By the above, it is sufficient to find $b_i \in \mathbf{Z}_p^\times$ such that $b_i^{p_i^{e_i}} = 1$ and $b_i^{p_i^{e_i-1}} \neq 1$.

We compute $b_{ij} \in \mathbf{Z}_p^\times$ for $j = 0, \ldots, e_i$ as follows. We set $b_{i0} = 1$ and, for $j \geq 1$, we factor the polynomial

$$x^{p_i} - b_{i,j-1} = \prod_{0 \leq k < p_i} (x - c_{ijk})$$

with $c_{ijk} \in Z_p^\times$, and set $b_{ij} = c_{ij0}$. (If $j = 1$, choose $b_{i1} \neq 1$.) Then $b_{ij}^{p_i^j} = 1$ for all j, and $b_{i,j-1}$ has a p_ith root by Corollary 2.3. Thus $x^{p_i} - b_{i,j-1}$ is indeed the product of linear polynomials, as claimed above. Also $b_{ij}^{p_i^{j-1}} \neq 1$ for $j \geq 1$, and $b_i = b_{i,e_i}$ is sufficient. □

Remark 5.2. It is not known whether there are infinitely many primes for which Theorem 5.1 is of interest, i.e., whether for infinitely many n there exist prime numbers p near 2^n, say $2^{n/2} < p < 2^n$, with all prime factors of $p-1$ at most n. Trivially, all primes $p \leq n$ have the latter property. The number of 'smooth' integers $k < 2^n$, i.e., with only small prime factors (but not $k+1$ necessarily prime), is well-studied, see, e.g., [21, 28].

If we assume the ERH, then we obtain polynomial-time algorithms for both our problems. Wang [33] shows that, under the ERH, there exists a primitive element a modulo p with $a = O((\log p)^8)$, for any prime p. If $p-1$ is smooth, primitivity can be tested fast, and we can find a primitive element in deterministic polynomial time. One can also use Ankeny's [3] theorem saying that, under the ERH, for every $i \leq r$, there exists some small $c_i \in N$ such that $c_i \not\equiv 0 \bmod p$ and $c_i^{(p-1)/p_i} \not\equiv 1$, namely $c_i = O(\log^2 p)$. (See [4] for a quantitative version of this theorem.) By exhaustive search, we find such a c_i, and set $b_i = c_i^{q_i}$. Then

$$b_i^{p_i^{e_i}} = c_i^{p-1} = 1, \qquad b_i^{p_i^{e_i-1}} = c_i^{(p-1)/p_i} \neq 1,$$

and $a = b_1 \cdots b_r$ is a primitive element modulo p.

Theorem 5.3. *Under the ERH,* FACTOR *and* PRIMITIVE *are in* P.

Recall that the running time of this algorithm for factoring a polynomial of degree d over $GF(p^l)$ is polynomial in $n = \max\{d, l, \log p, p_1, \ldots, p_r\}$, where $p_1, \ldots, p_r$ are the prime factors of $p-1$.

Remark 5.4. We want to discuss the question of fast parallel deterministic algorithms for factoring or for finding primitive elements. For us, 'fast parallel algorithm' means (log-space uniform families of) Boolean circuits of depth $(\log n)^{O(1)}$ and size $n^{O(1)}$, for input size n. See [13, 17] for general discussions of parallel Boolean and arithmetic computations.

Given a prime number $p < 2^n$, one can test whether all prime factors of $p-1$ are at most n, and, in the affirmative case, compute a factorization $p-1 = p_1^{e_1} \cdots p_r^{e_r}$ fast in parallel. It is not clear how to find a primitive element modulo p, and even the following is an open question: Given a prime p as above and $a \in N$ with $2 \leq a < p$, can one test fast in parallel whether a is a primitive element modulo p? Note that one cannot compute $a^{(p-1)/p_i}$ fast in parallel if only arithmetic modulo p is used [15].

Now consider the problem of factoring $f \in F[x]$ of degree d, with $F = GF(p^l)$. If $l = 1$, the square-free part of f can be computed in Boolean depth $O(\log^2(dl) + \log\log p \log\log\log p)$, which is poly-logarithmic in the input size (see [14]). In

general, f can be factored deterministically in Boolean depth $O(\log^2(dl)\log(dp)+\log^2 l \log\log l)$ [19], which is a 'fast parallel algorithm' only if p is small, e.g., when p is of order $(d+l)^{O(1)}$.

References

[1] L. Adleman, Two theorems on random polynomial time, *Proc. 19th Ann. IEEE Symp. on Foundations of Computer Science*, Ann Arbor, MI (1978) 75-83.

[2] L. Adleman and H.W. Lenstra, Finding irreducible polynomials over finite fields, *Proc. 18th Ann. ACM Symp. on Theory of Computing*, Berkeley, CA (1986) 350-355.

[3] N.C. Ankeny, The least quadratic non-residue, *Ann. of Math.* **55** (1952) 65-72.

[4] E. Bach, Fast algorithms under the Extended Riemann Hypothesis: a concrete estimate, *Proc. 14th Ann. ACM Symp. on Theory of Computing*, San Francisco, CA (1982) 290-295.

[5] E. Bach and J. Shallit, Factoring with cyclotomic polynomials, *Proc. 26th. Ann. IEEE Symp. on Foundations of Computer Science*, Portland, OR (1985) 443-450.

[6] M. Ben-Or, Probabilistic algorithms in finite fields, *Proc. 22nd IEEE Symp. on Foundations of Computer Science* (1981) 394-398.

[7] E.R. Berlekamp, Factoring polynomials over finite fields, *Bell System Tech. J.* **46** (1967) 1853-1859.

[8] E.R. Berlekamp, Factoring polynomials over large finite fields, *Math. Comp.* **24** (1970) 713-735.

[9] M.C.R. Butler, On the reducibility of polynomials over a finite field, *Quart. J. Math. Oxford* **5** (1954) 102-107.

[10] P. Camion, A deterministic algorithm for factorizing polynomials of $F_q[x]$, *Ann. Discr. Math.* **17** (1983) 149-157.

[11] D.G. Cantor and H. Zassenhaus, A new algorithm for factoring polynomials over finite fields, *Math. Comp.* **36** (1981) 587-592.

[12] A.L. Chistov and D.Yu. Grigoryev, Polynomial-time factoring of the multivariable polynomials over a global field, LOMI preprint E-5-82, Leningrad, 1982.

[13] S.A. Cook, A taxonomy of problems with fast parallel algorithms, *Inform. and Control* **64** (1985) 2-22.

[13a] D. Coppersmith and S. Winograd, Matrix multiplication via Behrend's Theorem, *Proc. 19th Ann. ACM Symp. on Theory of Computing*, New York, NY (1987).

[14] J. von zur Gathen, Parallel algorithms for algebraic problems, *SIAM J. Comput.* **13** (1984) 802-824.

[15] J. von zur Gathen, Parallel powering, *Proc. 25th IEEE Symp. on Foundations of Computer Science*, Singer Island, FL (1984) 31-36; also *SIAM J. Comput.*, to appear.

[16] J. von zur Gathen, Irreducible polynomials over finite fields, *Proc. 6th Conf. on Foundations of Software Technology and Theoretical Computer Science*, Delhi, India, Lecture Notes in Computer Science **241** (Springer, Berlin, 1986) 252-262.

[17] J. von zur Gathen, Parallel arithmetic computations: a survey, in: *Proc. 12th Internat. Symp. on Math. Foundations of Computer Science*, Bratislava, Lecture Notes in Computer Science **233** (Springer, Berlin, 1986) 93-112.

[18] J. von zur Gathen and E. Kaltofen, Factorization of multivariate polynomials over finite fields, *Math. Comp.* **45** (1985) 251-261.

[19] J. von zur Gathen and G. Seroussi, Boolean circuits versus arithmetic circuits, *Proc. 6th Internat. Conf. on Computer Science*, Santiago, Chile (1986) 171-184.

[20] H. Gunji and D. Arnon, On polynomial factorization over finite fields, *Math. Comp.* **36** (1981) 281-287.

[21] A. Hildebrand, On the number of positive integers $\leq x$ and free of prime factors $>y$, *J. Number Theory* **22** (1986) 289-307.

[22] M.A. Huang, Riemann Hypothesis and finding roots over finite fields, *Proc. 17th Ann. ACM Symp. on Theory of Computing*, Providence, RI (1985) 121-130.

[23] E. Kaltofen, Uniform closure properties of p-computable functions, *Proc. 18th Ann. ACM Symp. on Theory of Computing*, Berkeley, CA (1986) 330-337.

[24] D.E. Knuth, *The Art of Computer Programming, Vol. 2* (Addison-Wesley, Reading, MA, 1981, 2nd ed.).

[25] A.K. Lenstra, Factoring multivariate polynomals over finite fields, *J. Comput. System Sci.* **30** (1985) 235-248.

[26] A.K. Lenstra, H.W. Lenstra and L. Lovász, Factoring polynomials with rational coefficients, *Math. Ann.* **261** (1982) 515-534.

[27] R.T. Moenck, On the efficiency of algorithms for polynomial factoring, *Math. Comp.* **31** (1977) 235-250.

[28] K. Norton, Numbers with small prime factors and the least *k*th power non-residue, *Mem. Amer. Math. Soc.* **106** (1971).

[29] M.O. Rabin, Probabilistic algorithms in finite fields, *SIAM J. Comput.* **9** (1980) 273-280.

[30] A. Schönhage, Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2, *Acta Inform.* **7** (1977) 395-398.

[31] A. Schönhage and V. Strassen, Schnelle Multiplikation grosser Zahlen, *Computing* **7** (1971) 281-292.

[32] R.J. Schoof, Elliptic curves over finite fields and the computation of square roots mod *p*, *Math. Comp.* **44** (1985) 483-494.

[33] Y. Wang, On the least primitive root of a prime, *Acta Math. Sinica* **9** (1959) 432-441; English translation in: *Scientia Sinica* **10** (1961) 1-14.